RGCL at GermEval 2019: Offensive Language Detection with Deep Learning

Alistair Plum, Tharindu Ranasinghe, Constantin Orăsan, Ruslan Mitkov
Research Group in Computational Linguistics
University of Wolverhampton, UK
{a.j.plum, tharindu.ranasinghe, c.orasan, r.mitkov}@wlv.ac.uk

Abstract

This paper describes the system submitted by the RGCL team to *GermEval 2019 Shared Task 2: Identification of Offensive Language*. We experimented with five different neural network architectures in order to classify Tweets in terms of offensive language. By means of comparative evaluation, we select the best performing for each of the three subtasks. Overall, we demonstrate that using only minimal preprocessing we are able to obtain competitive results.

1 Introduction

The use of offensive language in open and public actions is a facet of the internet that calls for automatic detection for some kind of monitoring. The fact that people use language that can cause offence to other people is in no way a novel phenomenon. However, with the rise of online platforms such as Twitter, Facebook, Reddit and so on, along with the annonymity these platforms offer, offensive language can be viewed and read by millions of people in an instance. While the scale of the offence caused by such language can vary, it is clear that there is some language which causes offence to many people publicly. Therefore, it is desirable to be able to automatically detect the use of such language, in order to flag it and take further action.

Recently, efforts in the field of natural language processing (NLP) relating to the detection and classification of offensive language have been gaining attention. This is not only evidenced by an increase in offensive language datasets, but also a shift in approach from support vector machine classifiers to more modern neural networks (Schmidt and Wiegand, 2017). More evidence for the rising attention to offensive language detection lies in the fact that the topic has been featured at shared tasks. The most prominent example is probably SemEval 2019 Task 6, which dealt with the identification and categorisation of offensive language in social media for English, and attracted around 800 teams with 115 submissions (Zampieri et al., 2019).

Another example of a shared task for offensive language is GermEval 2019 Task 2. It deals with the detection and classification of offensive language in German Twitter posts. The task itself is divided into three classification subtasks, with the first dealing with a binary classification, i.e. whether a tweet is offensive or not. Subtask II is a more fine-grained classification, including three levels of granularity, *profanity*, *insult* and *abuse*. These two subtasks were featured at GermEval 2018, meaning that data from two years was available (Wiegand et al., 2018). The final subtask, aimed at classifying implicit and explicit offensive language, was newly introduced this year.

This paper describes our submission to the GermEval shared task. We propose a simple, low-effort approach, with minimal data processing. We employ five different neural network architectures in order to perform the three classification subtasks, evaluate each network and select the three best performing architectures for our final submissions.

The paper is structured as follows. Section 2 describes the system that was submitted, split into a description of the dataset (Section 2.1), how the data was processed (Section 2.2) and the architecture of the classifier that was used (Section 2.3). Section 3 presents an analysis of the results of our evaluation of the five different architectures (Section 3.1), as well as of the final submission (Section 3.2). Finally, Section 4 offers some final remarks and a conclusion.

2 System Description

This section describes the shared task data, as well as the system that was used to classify the data. The dataset is grouped in two parts, and we use

minimal preprocessing in order to use the data. For classification, we used and compared five different neural network architectures suited to this task. Our implementation has been made available on Github.[1]

2.1 Dataset

The data provided by the task organisers was split into subtasks I & II, and subtask III, which where in turn split into training and test sets. For subtasks I & II we concatenated the 2018 training set (Wiegand et al., 2018) with the 2019 training set, resulting in 9004 training instances for subtasks I & II. Subtask III was introduced for the first time this year, providing 1958 training instances.

Different tags are used for each subtask. The binary classification uses *OTHER* and *OFFENSIVE* to distinguish non-offensive and offensive text. Subtask II extends the last tag into *PROFANITY* indicating the use of offensive words without being aimed at anyone specific, *INSULT* which is like the previous but aimed at a specific person or entity, and *ABUSE* which combines the last two tags. The final subtask uses the *EXPLICIT* and *IMPLICIT* tags.

2.2 Text Preprocessing

As mentioned previously, the data preprocessing for this task was kept fairly minimal. More specifically, we perform only three specialised tasks for this data, followed by tokenisation. The tasks include removing usernames, converting to lower case and removing punctuation marks. The motivation for a minimal approach was mainly to demonstrate the effectiveness of the neural network architectures used. A secondary motivation is the portability to other languages, as the tasks carried out here should be relatively simple to perform in other languages. This does, however, highlight the importance of solid word embeddings, as the approach is completely reliant on them.

First, we completely remove all usernames from the texts, without inserting a placeholder. This is carried out quite simply by removing all strings beginning with the @ symbol, as this is how usernames are denoted on Twitter. The reasoning behind this step is mainly to remove noisy text, as it is highly unlikely that there would be any embeddings for the usernames. In addition, it stands to reason that these usernames do not add any semantic meaning. Moreover, if, for instance, a majority of offensive tweets were written by one user, this could lead to bias in the system against one user. However, this task is targeted at offensive language, not offensive users.

Next, we convert the text to lower case letters. This step may seem counter intuitive for German, as capitalisation is used to differentiate nouns, which can cause a difference in meaning. For instance, *Rennen* can mean *the race* (noun), whereas *rennen* can mean *to run* (verb). Therefore, our first intuition was to keep capitalisation, however, after running with and without capitalisation our results indicated that all lower case text works better. We found that this leads to a smaller number of words for which no embedding is found, and higher precision and recall values. This finding is in line with previous findings using a similar approach based on neural networks (Stammbach et al., 2018).

Finally, we remove all kinds of punctuation marks and mathematical symbols. We insert a place-holder that indicates to the system that a punctuation mark would have been at this place. With this approach, we can handle each word in the same way, as leaving punctuation marks would lead them to be read together with a word, leading to no embedding being available.

2.3 System Architecture

After data processing each text is encoded using German fasttext (Mikolov et al., 2018) embeddings.[2] The encoded tweets are then classified by one of the neural network architectures. We evaluated five different neural network architectures for the classification tasks: pooled Gated Recurrent Unit (GRU) (Section 2.3.1), Long Short-Term Memory (LSTM) and GRU with Attention (Section 2.3.2), 2D Convolution with Pooling (Section 2.3.3), GRU with Capsule (Section 2.3.4) and LSTM with Capsule and Attention (Section 2.3.5).

The parameters of each architecture were optimised using 5-fold cross-validation considering binary cross entropy loss function and using adam optimiser (Kingma and Ba, 2015). The motivation for using 5-fold cross-validation was mainly the size of the data available for subtask III. Using a higher number of folds for cross-validation results in a low number of training and evaluation

[1] https://github.com/TharinduDR/Germeval-Task-2

[2] https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.de.vec

instances affects the performance of the architecture (Stone, 1974). We used the reducing learning rate on plateau technique when a deep learning architecture stopped improving. Deep learning architectures often benefit from reducing the learning rate by a factor once learning stagnates (Ravaut and Gorti, 2018). We monitored validation loss and if no improvement was seen for 2 epochs, the learning rate was reduced by a factor of 0.6, since this value seemed to offer the best improvement.

These architectures were successfully applied to a number of classification tasks such as GRU for sequence labeling (Chung et al., 2014), GRU with capsule for toponym detection (Plum et al., 2019), and their success in these tasks inspired us to use them for the task at hand.

2.3.1 Pooled GRU

In this architecture, after the embedding layer, embedding vectors are fed to the bi-directional GRU (Chung et al., 2014) at their respective timestep. The bi-directional GRU-layer has 80 units. The final timestep output is fed into a max pooling layer and an average pooling layer in parallel (Scherer et al., 2010). After this, the outputs of the two pooling layers are concatenated and connected to a dense layer (Huang et al., 2017) activated with a sigmoid function. Additionally, there is a spatial dropout (Tompson et al., 2015) between the embedding layer and the bi-directional GRU layer to avoid over-fitting. This architecture has been discussed in (Kowsari et al., 2019) as a common architecture to perform text classification tasks.

2.3.2 LSTM and GRU with Attention

With this architecture, the output of the embedding layer goes through a spatial dropout (Tompson et al., 2015) and is then fed in parallel to a bi-directional LSTM-layer (Schuster and Paliwal, 1997) with self attention and a bi-directional GRU-layer (Chung et al., 2014) with self attention (Vaswani et al., 2017). Both the bi-directional LSTM-layer and the bi-directional GRU-layer have 40 units. The output from the bi-directional GRU-layer is fed into an average pooling layer and a max pooling layer. The output from these layers and the output of the bi-directional LSTM-layer are concatenated and connected to a dense layer with ReLU activation. After that, a dropout (Srivastava et al., 2014) is applied to the output and connected to a dense layer activated with a sigmoid function.

2.3.3 2D Convolution with Pooling

The fourth architecture takes a different approach than the previous architectures by using 2D convolution layers (Wu et al., 2018), rather than LSTM or GRU layers. The outputs of the embedding layers are connected to four 2D convolution layers (Wu et al., 2018), each with max pooling layers. All the 2D convolution layers were initialised with normal kernel initialiser. The outputs of these are concatenated and connected to a dense layer activated with a sigmoid function after applying a dropout (Srivastava et al., 2014). This architecture has been used in the Quora Insincere Questions Classification Kaggle competition[3].

2.3.4 GRU with Capsule

Most of the previous architectures rely on a pooling layer. However, this architecture uses a capsule layer (Hinton et al., 2018) rather than pooling layers. After applying a spatial dropout (Tompson et al., 2015) the output of the embedding layer is fed into a bi-directional GRU-layer (Chung et al., 2014). The bi-directional GRU-layer has 100 units and was initialised with the Glorot normal kernel initialiser and orthogonal recurrent initialiser with 1.0 gain. The output is then connected to a capsule layer (Hinton et al., 2018). The output of the capsule layer is flattened and connected to a dense layer with ReLU activation, a dropout (Srivastava et al., 2014) and batch normalisation applied, and re-connected to a dense layer with sigmoid activation. This architecture has been used to detect locations within word windows (Plum et al., 2019).

2.3.5 LSTM with Capsule and Attention

The final architecture uses combination of a capsule layer (Hinton et al., 2018) and a self attention layer (Vaswani et al., 2017). After the embedding layer a spatial dropout (Tompson et al., 2015) is applied to the output, which is then fed into a bi-directional LSTM-layer (Schuster and Paliwal, 1997) with 80 units. The layer is initialised with the Glorot normal kernel initialiser and orthogonal recurrent initialiser with 1.0 gain. The output of the bi-directional LSTM-layer is fed into a capsule layer and to a self attention layer in parallel. Then each output of both capsule layers and the self attention layer goes through a DropConnect (Wan et al., 2013). They are concatenated before connecting

[3] https://www.kaggle.com/c/quora-insincere-questions-classification

to a dense layer with sigmoid activation. This architecture has been used in the *Jigsaw Unintended Bias in Toxicity Classification* competition.[4]

3 Results

This section presents the results of the evaluation of the five architectures, as well as the evaluation of the final submission. As outlined in the previous sections, we compare the performance of five different neural network architectures in order to select the best for each task. Therefore, an evaluation of each architecture was performed, the results of which are presented in Section 3.1. In Section 3.2 we present the results of the final submission as carried out by the organisers of the task. Although we submitted the runs of the three best performing systems, we only present the best performing here. The full results have been added to Appendix A.

3.1 Architecture Evaluation

This section describes how we selected the architectures for the final submissions in each subtask. For subtask I, all of the architectures were trained on the 2018 and 2019 training data. The architectures were evaluated on the 2018 test data. GRU with Capsule, 2D Convolution with Pooling and Pooled GRU had the best F1-scores with 0.743, 0.740 and 0.728, respectively.

Again, for subtask II all of the architectures were trained on the 2018 and 2019 training data, and evaluated on the 2018 test data. GRU with Capsule, 2D Convolution with Pooling and LSTM & GRU with Attention were selected for final submission, with F1-scores of 0.698, 0.695 and 0.684, respectively.

As subtask III was organised for the first time this year, we did not have 2018 training data for architecture training or 2018 testing data for evaluation. Nonetheless, for subtask III we used 20% of the available 2019 training data for the evaluation, and used the rest of the data for training. For subtask III, GRU with Capsule, Pooled GRU and 2D Convolution with Pooling were used for the final submission, as they had F1-scores of 0.887, 0.840 and 0.817, respectively.

It is interesting to note that the GRU with Capsule and 2D Convolution with Pooling architectures were always among the top three performing architectures.

[4] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

Subtask	P	R	F1	Acc.
I	79.49	67.94	73.26	77.96
II	58.64	36.53	45.02	72.35
III	65.55	72.55	68.87	80.11

Table 1: Results of the evaluation. All values are reported as percent.

3.2 Submission Results

This section presents the results of the evaluation of our submission. The evaluation was carried out by the task organisers, and at the time of writing the paper the results and rankings of other groups are not available. Therefore, we report only the evaluation provided to us by the task organisers. We report precision, recall and f-measure averaged overall for each classification subtask. Separate values for each group of each individual classification task are presented in the full results, as well as the results for the other two architectures. Table 1 shows the results of the evaluation of the best performing architecture, 2D Convolution with Pooling.

4 Conclusion

In this paper, we have presented our system for identifying offensive language in tweets. The system uses minimal preprocessing, and relies on word embeddings. We experimented with different neural network architectures in order to determine the most suitable for this task. Going by our evaluation, and the results provided by the task organisers, it is clear that 2D Convolution with Pooling scores highest overall.

While our system should be quite portable to other languages, due to non language-specific preprocessing, it is also clear that this aspect could potentially improve the performance of our system. Moreover, a detailed look into the results of the fine-grained classification of subtask II could yield good indications of how to improve the system for this kind of classification. Nonetheless, for future research we would like to see how well this system could perform in other languages on similar tasks.

References

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*.

Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *Proceedings of ICLR 2018*.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *Proceedings of IEEE CVPR 2017*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of CoRR 2015*.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text Classification Algorithms: A Survey. *Information*, 10(4).

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of LREC 2018*.

Alistair Plum, Tharindu Ranasinghe, Pablo Calleja, Constantin Orasan, and Ruslan Mitkov. 2019. RGCL-WLV at SemEval-2019 Task 12: Toponym Detection. In *Proceedings of SemEval-2019*.

Mathieu Ravaut and Satya Gorti. 2018. Gradient descent revisited via an adaptive online learning rate. *arXiv preprint arXiv:1801.09136*.

Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of ICANN 2010*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for GermEval task 2018. In *Proceedings of GermEval 2018*.

Mervyn Stone. 1974. Cross-validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using Convolutional Networks. In *Proceedings of IEEE CVPR 2015*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS*.

Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of Neural Networks using DropConnect. In *Proceedings of ICML 2013*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval 2018*.

Yunan Wu, Feng Yang, Ying Liu, Xuefan Zha, and Shaofeng Yuan. 2018. A Comparison of 1-D and 2-D Deep Convolutional Neural Networks in ECG Classification. In *Proceedings of IEEE Engineering in Medicine and Biology Society*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). *CoRR*.

Subtask I

	Accuracy			OFFENSE			OTHER			Average		
File	Percent	correct	total	P	R	F	P	R	F	P	R	F-Score
2	77.96	2363	3031	81.72	40.1	53.8	77.26	95.78	85.53	79.49	67.94	**73.26**
3	77.37	2345	3031	83.49	36.49	50.79	76.37	96.6	85.3	79.93	66.55	**72.63**
1	75.92	2301	3031	82.79	31.24	45.36	74.97	96.94	84.55	78.88	64.09	**70.72**

Subtask II

	Accuracy			ABUSE			INSULT		
File	Percent	correct	total	P	R	F	P	R	F
2	72.35	2193	3031	49.04	25.5	33.55	52.84	20.26	29.29
1	71.56	2169	3031	45.66	25	32.31	54.3	17.86	26.89
3	72.45	2196	3031	48.53	24.75	32.78	50.28	19.39	27.99

	PROFANITY			OTHER			Average		
	P	R	F	P	R	F	P	R	F-Score
	57.14	3.6	6.78	75.53	96.75	84.83	58.64	36.53	**45.02**
	52.94	8.11	14.06	74.81	95.97	84.08	56.93	36.74	**44.66**
	0	0	0	75.77	97.43	85.25	43.65	35.39	**39.09**

Subtask III

	Accuracy			EXPLICIT			IMPLICIT			Average		
File	Percent	correct	total	P	R	F	P	R	F	P	R	F-Score
3	80.11	745	930	92.85	83.17	87.74	38.25	61.94	47.29	65.55	72.55	**68.87**
1	78.39	729	930	93.05	80.78	86.48	35.98	64.18	46.11	64.52	72.48	**68.27**
2	78.17	727	930	91.24	82.41	86.6	33.65	52.99	41.16	62.44	67.7	**64.96**